A New Statistical Parser Based on Bigram Lexical Dependencies

Michael John Collins*
Dept. of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, 19104, U.S.A.
mcollins@gradient.cis.upenn.edu

Abstract

This paper describes a new statistical parser which is based on probabilities of dependencies between head-words in the parse tree. Standard bigram probability estimation techniques are extended to calculate probabilities of dependencies between pairs of words. Tests using Wall Street Journal data show that the method performs at least as well as SPATTER (Magerman 95; Jelinek et al. 94), which has the best published results for a statistical parser on this task. The simplicity of the approach means the model trains on 40,000 sentences in under 15 minutes. With a beam search strategy parsing speed can be improved to over 200 sentences a minute with negligible loss in accuracy.

1 Introduction

Lexical information has been shown to be crucial for many parsing decisions, such as prepositional-phrase attachment (for example (Hindle and Rooth 93)). However, early approaches to probabilistic parsing (Pereira and Schabes 92; Magerman and Marcus 91; Briscoe and Carroll 93) conditioned probabilities on non-terminal labels and part of speech tags alone. The SPATTER parser (Magerman 95; Jelinek et al. 94) does use lexical information, and recovers labeled constituents in Wall Street Journal text with above 84% accuracy – as far as we know the best published results on this task.

This paper describes a new parser which is much simpler than SPATTER, yet performs at least as well when trained and tested on the same Wall Street Journal data. The method uses lexical information directly by modeling head-modifier[1] relations between pairs of words. In this way it is similar to Link grammars (Lafferty et al. 92), and dependency grammars in general.

2 The Statistical Model

The aim of a parser is to take a tagged sentence as input (for example Figure 1(a)) and produce a phrase-structure tree as output (Figure 1(b)). A statistical approach to this problem consists of two components. First, the *statistical model* assigns a probability to every candidate parse tree for a sentence. Formally, given a sentence S and a tree T, the model estimates the conditional probability $P(T|S)$. The most likely parse under the model is then:

$$T_{best} = argmax_T P(T|S) \quad (1)$$

Second, the *parser* is a method for finding T_{best}. This section describes the statistical model, while section 3 describes the parser.

The key to the statistical model is that any tree such as Figure 1(b) can be represented as a set of **baseNPs**[2] and a set of **dependencies** as in Figure 1(c). We call the set of baseNPs B, and the set of dependencies D; Figure 1(d) shows B and D for this example. For the purposes of our model, $T = (B, D)$, and:

$$P(T|S) = P(B, D|S) = P(B|S) \times P(D|S, B) \quad (2)$$

S is the sentence with words tagged for part of speech. That is, $S = <(w_1, t_1), (w_2, t_2) ... (w_n, t_n)>$. For POS tagging we use a maximum-entropy tagger described in (Ratnaparkhi 96). The tagger performs at around 97% accuracy on Wall Street Journal Text, and is trained on the first 40,000 sentences of the Penn Treebank (Marcus et al. 93).

Given S and B, the **reduced sentence** $\bar{S}$ is defined as the subsequence of S which is formed by removing punctuation and reducing all baseNPs to their head-word alone.

*This research was supported by ARPA Grant N6600194-C6043.

[1] By 'modifier' we mean the linguistic notion of either an argument or adjunct.

[2] A baseNP or 'minimal' NP is a non-recursive NP, i.e. none of its child constituents are NPs. The term was first used in (Ramshaw and Marcus 95).

(a)

John/NNP Smith/NNP, the/DT president/NN of/IN IBM/NNP, announced/VBD his/PRP$ resignation/NN yesterday/NN .

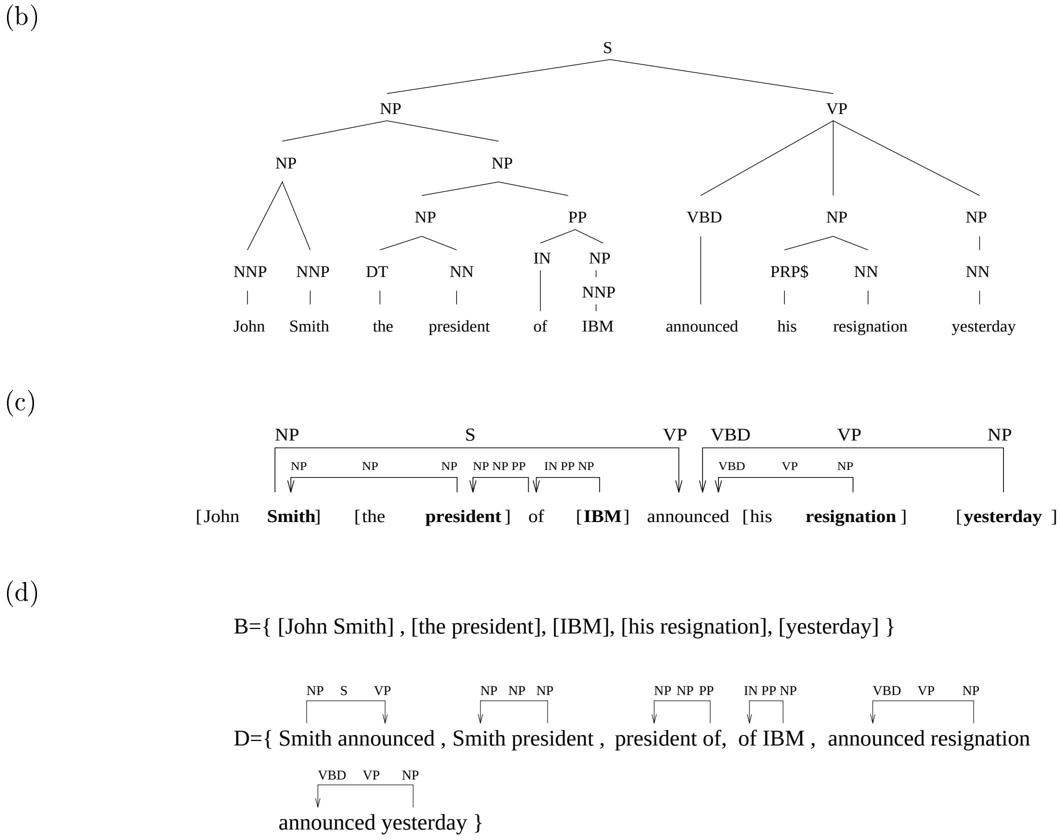

Figure 1: An overview of the representation used by the model. **(a)** The tagged sentence; **(b)** A candidate parse-tree (the correct one); **(c)** A dependency representation of (b). Square brackets enclose baseNPs (heads of baseNPs are marked in bold). Arrows show modifier → head dependencies. Section 2.1 describes how arrows are labeled with non-terminal triples from the parse-tree. Non-head words within baseNPs are excluded from the dependency structure; **(d)** B, the set of baseNPs, and D, the set of dependencies, are extracted from (c).

Thus the reduced sentence is an array of word/tag pairs, $\bar{S} =< \langle \bar{w}_1, \bar{t}_1 \rangle, \langle \bar{w}_2, \bar{t}_2 \rangle ... \langle \bar{w}_m, \bar{t}_m \rangle >$, where $m \leq n$. For example for Figure 1(a)

Example 1 $\bar{S} =$

$$< \langle Smith, NNP \rangle, \langle president, NN \rangle, \langle of, IN \rangle, \langle IBM, NNP \rangle, \langle announced, VBD \rangle, \langle resignation, NN \rangle, \langle yesterday, NN \rangle >$$

Sections 2.1 to 2.4 describe the dependency model. Section 2.5 then describes the baseNP model, which uses bigram tagging techniques similar to (Ramshaw and Marcus 95; Church 88).

2.1 The Mapping from Trees to Sets of Dependencies

The dependency model is limited to relationships between words in **reduced** sentences such as Example 1. The mapping from trees to dependency structures is central to the dependency model. It is defined in two steps:

1. For each constituent $P \rightarrow < C_1 ... C_n >$ in the parse tree a simple set of rules[3] identifies which of the children C_i is the 'head-child' of P. For example, `NN` would be identified as the head-child of `NP` → `<DET JJ JJ NN>`, `VP` would be identified as the head-child of `S` → `<NP VP>`. Head-words propagate up through the tree, each parent receiving its head-word from its head-child. For example, in `S` → `<NP VP>`, `S` gets its head-word, *announced*,

[3] The rules are essentially the same as in (Magerman 95; Jelinek et al. 94). These rules are also used to find the head-word of baseNPs, enabling the mapping from S and B to $\bar{S}$.

from its head-child, the `VP`.

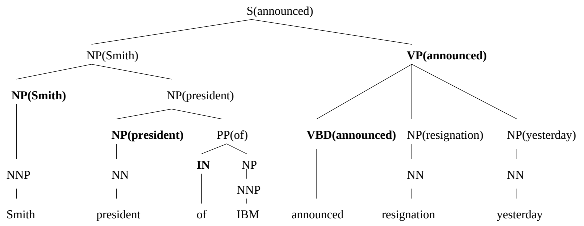

Figure 2: Parse tree for the reduced sentence in Example 1. The head-**child** of each constituent is shown in bold. The head-**word** for each constituent is shown in parentheses.

2. Head-modifier relationships are now extracted from the tree in Figure 2. Figure 3 illustrates how each constituent contributes a set of dependency relationships. `VBD` is identified as the head-child of `VP` $\rightarrow$ <`VBD NP NP`>. The head-words of the two NPs, *resignation* and *yesterday*, both modify the head-word of the `VBD`, *announced*. Dependencies are labeled by the modifier non-terminal, `NP` in both of these cases, the parent non-terminal, `VP`, and finally the head-child non-terminal, `VBD`. The triple of non-terminals at the start, middle and end of the arrow specify the nature of the dependency relationship – <`NP,S,VP`> represents a subject-verb dependency, <`PP,NP,NP`> denotes prepositional phrase modification of an `NP`, and so on[4].

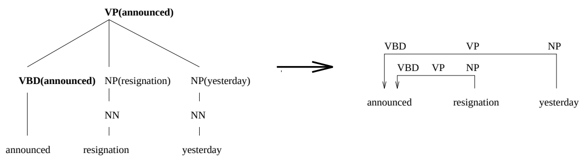

Figure 3: Each constituent with n children (in this case $n = 3$) contributes $n - 1$ dependencies.

Each word in the reduced sentence, with the exception of the sentential head 'announced', modifies exactly one other word. We use the notation

$$AF(j) = (h_j, R_j) \tag{3}$$

to state that the jth word in the reduced sentence is a modifier to the h_jth word, with relationship R_j[5]. AF stands for 'arrow from'. R_j is the triple of labels at the start, middle and end of the arrow. For example, $\bar{w}_1 = Smith$ in this sentence, and $\bar{w}_5 = announced$, so $AF(1) = (5, \texttt{<NP,S,VP>})$.

D is now defined as the m-tuple of dependencies: $D = \{(AF(1), AF(2)...AF(m)\}$. The model assumes that the dependencies are independent, so that:

$$P(D|S,B) = \prod_{j=1}^{m} P(AF(j)|S,B) \tag{4}$$

2.2 Calculating Dependency Probabilities

This section describes the way $P(AF(j)|S,B)$ is estimated. The same sentence is very unlikely to appear both in training and test data, so we need to back-off from the entire sentence context. We believe that lexical information is crucial to attachment decisions, so it is natural to condition on the words and tags. Let $\mathcal{V}$ be the vocabulary of all words seen in training data, $\mathcal{T}$ be the set of all part-of-speech tags, and $\mathcal{TRAIN}$ be the training set, a set of reduced sentences. We define the following functions:

- $C\,(\,\langle a,b\rangle\,,\,\langle c,d\rangle\,)$ for $a, c \,\epsilon\, \mathcal{V}$, and $b, d \,\epsilon\, \mathcal{T}$ is the number of times $\langle a,b\rangle$ and $\langle c,d\rangle$ are seen in the same reduced sentence in training data.[6] Formally,

$$C\,(\,\langle a,b\rangle\,,\,\langle c,d\rangle\,) = \sum_{\substack{\bar{S}\,\epsilon\,\mathcal{TRAIN} \\ k,l=1..|\bar{S}|,\; l\neq k}} h\left(\bar{S}[k] = \langle a,b\rangle\,,\,\bar{S}[l] = \langle c,d\rangle\right) \tag{5}$$

where $h(x)$ is an indicator function which is 1 if x is true, 0 if x is false.

- $C\,(R,\,\langle a,b\rangle\,,\,\langle c,d\rangle\,)$ is the number of times $\langle a,b\rangle$ and $\langle c,d\rangle$ are seen in the same reduced sentence in training data, and $\langle a,b\rangle$ modifies $\langle c,d\rangle$ with relationship R. Formally,

$$C\,(R,\,\langle a,b\rangle\,,\,\langle c,d\rangle\,) = \sum_{\substack{\bar{S}\,\epsilon\,\mathcal{TRAIN} \\ k,l=1..|\bar{S}|,\; l\neq k}} h(\bar{S}[k] = \langle a,b\rangle\,,\,\bar{S}[l] = \langle c,d\rangle\,,\,AF(k) = (l,R)) \tag{6}$$

- $F(R\,|\,\langle a,b\rangle\,,\,\langle c,d\rangle\,)$ is the probability that $\langle a,b\rangle$ modifies $\langle c,d\rangle$ with relationship R, given that $\langle a,b\rangle$ and $\langle c,d\rangle$ appear in the same reduced sentence. The maximum-likelihood estimate of $F(R\,|\,\langle a,b\rangle\,,\,\langle c,d\rangle\,)$ is:

$$\hat{F}(R\,|\,\langle a,b\rangle\,,\,\langle c,d\rangle\,) = \frac{C(R,\,\langle a,b\rangle\,,\,\langle c,d\rangle\,)}{C(\,\langle a,b\rangle\,,\,\langle c,d\rangle\,)} \tag{7}$$

We can now make the following approximation:

$$P(AF(j) = (h_j, R_j)\mid S,B) \approx \frac{\hat{F}(R_j\,|\,\langle \bar{w}_j, \bar{t}_j\rangle\,,\,\langle \bar{w}_{h_j}, \bar{t}_{h_j}\rangle\,)}{\sum_{k=1..m,k\neq j,p\epsilon\mathcal{P}} \hat{F}(p\,|\,\langle \bar{w}_j, \bar{t}_j\rangle\,,\,\langle \bar{w}_k, \bar{t}_k\rangle\,)} \tag{8}$$

[4]The triple can also be viewed as representing a semantic predicate-argument relationship, with the three elements being the type of the argument, result and functor respectively. This is particularly apparent in Categorial Grammar formalisms (Wood 93), which make an explicit link between dependencies and functional application.

[5]For the head-word of the entire sentence $h_j = 0$, with R_j=<Label of the root of the parse tree >. So in this case, $AF(5) = (0, < S >)$.

[6]Note that we count multiple co-occurrences in a single sentence, e.g. if $\bar{S} = (< a,b >, < c,d >, < c,d >)$ then $C(< a,b >, < c,d >) = C(< c,d >, < a,b >) = 2$.

where $\mathcal{P}$ is the set of all triples of non-terminals. The denominator is a normalising factor which ensures that

$$\sum_{k=1..m,k\neq j,p\epsilon\mathcal{P}} P(AF(j) = (k,p) \mid S,B) = 1$$

From (4) and (8):

$$P(D|S,B) \approx \qquad (9)$$
$$\prod_{j=1}^{m} \frac{\hat{F}(R_j \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle)}{\sum_{k=1..m,k\neq j,p\epsilon\mathcal{P}} \hat{F}(p \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_k, \bar{t}_k \rangle)}$$

The denominator of (9) is constant, so maximising $P(D|S,B)$ over D for fixed S, B is equivalent to maximising the product of the numerators, $\mathcal{N}(D|S,B)$. (This considerably simplifies the parsing process):

$$\mathcal{N}(D|S,B) = \prod_{j=1}^{m} \hat{F}(R_j \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle) \quad (10)$$

2.3 The Distance Measure

An estimate based on the identities of the two tokens alone is problematic. Additional context, in particular the relative order of the two words and the distance between them, will also strongly influence the likelihood of one word modifying the other. For example consider the relationship between 'sales' and the three tokens of 'of':

Example 2 *Shaw, based in Dalton, Ga., has annual* **sales of** *about \$ 1.18 billion, and has economies* **of** *scale and lower raw-material costs that are expected to boost the profitability* **of** *Armstrong 's brands, sold under the Armstrong and Evans-Black names .*

In this sentence 'sales' and 'of' co-occur three times. The parse tree in training data indicates a relationship in only one of these cases, so this sentence would contribute an estimate of $\frac{1}{3}$ that the two words are related. This seems unreasonably low given that 'sales of' is a strong collocation. The latter two instances of 'of' are so distant from 'sales' that it is unlikely that there will be a dependency.

This suggests that distance is a crucial variable when deciding whether two words are related. It is included in the model by defining an extra 'distance' variable, Δ, and extending C, F and $\hat{F}$ to include this variable. For example, $C(\langle a,b \rangle, \langle c,d \rangle, \Delta)$ is the number of times $\langle a,b \rangle$ and $\langle c,d \rangle$ appear in the same sentence at a distance Δ apart. (11) is then maximised instead of (10):

$$\mathcal{N}(D|S,B) = \prod_{j=1}^{m} \hat{F}(R_j \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \quad (11)$$

A simple example of Δ_{j,h_j} would be $\Delta_{j,h_j} = h_j - j$. However, other features of a sentence, such as punctuation, are also useful when deciding if two words are related. We have developed a heuristic 'distance' measure which takes several such features into account The current distance measure Δ_{j,h_j} is the combination of 6 features, or questions (we motivate the choice of these questions qualitatively – section 4 gives quantitative results showing their merit):

Question 1 Does the h_jth word precede or follow the jth word? English is a language with strong word order, so the order of the two words in surface text will clearly affect their dependency statistics.

Question 2 Are the h_jth word and the jth word adjacent? English is largely right-branching and head-initial, which leads to a large proportion of dependencies being between adjacent words [7]. Table 1 shows just how local most dependencies are.

Distance	1	≤ 2	≤ 5	≤ 10
Percentage	74.2	86.3	95.6	99.0

Table 1: Percentage of dependencies vs. distance between the head words involved. These figures count baseNPs as a single word, and are taken from WSJ training data.

Number of verbs	0	<=1	<=2
Percentage	94.1	98.1	99.3

Table 2: Percentage of dependencies vs. number of verbs between the head words involved.

Question 3 Is there a verb between the h_jth word and the jth word? Conditioning on the exact distance between two words by making $\Delta_{j,h_j} = h_j - j$ leads to severe sparse data problems. But Table 1 shows the need to make finer distance distinctions than just whether two words are adjacent. Consider the prepositions 'to', 'in' and 'of' in the following sentence:

Example 3 *Oil stocks* **escaped** *the brunt of Friday 's selling and several were able to post gains , including Chevron , which* **rose** *5/8* **to** *66 3/8* **in** *Big Board composite trading* **of** *2.4 million shares .*

The prepositions' main candidates for attachment would appear to be the previous verb, 'rose', and the baseNP heads between each preposition and this verb. They are less likely to modify a more distant verb such as 'escaped'. Question 3 allows the parser to prefer modification of the most recent verb – effectively another, weaker preference for right-branching structures. Table 2 shows that 94% of dependencies do not cross a verb, giving empirical evidence that question 3 is useful.

[7] For example in '(John (likes (to (go (to (University (of Pennsylvania)))))))' all dependencies are between adjacent words.

Questions 4, 5 and 6

- Are there 0, 1, 2, or more than 2 'commas' between the h_jth word and the jth word? (All symbols tagged as a ',' or ':' are considered to be 'commas').
- Is there a 'comma' immediately following the first of the h_jth word and the jth word?
- Is there a 'comma' immediately preceding the second of the h_jth word and the jth word?

People find that punctuation is extremely useful for identifying phrase structure, and the parser described here also relies on it heavily. Commas are not considered to be words or modifiers in the dependency model – but they do give strong indications about the parse structure. Questions 4, 5 and 6 allow the parser to use this information.

2.4 Sparse Data

The maximum likelihood estimator in (7) is likely to be plagued by sparse data problems – $C(\langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j})$ may be too low to give a reliable estimate, or worse still it may be zero leaving the estimate undefined. (Collins and Brooks 95) describes how a backed-off estimation strategy is used for making prepositional phrase attachment decisions. The idea is to back-off to estimates based on less context. In this case, less context means looking at the POS tags rather than the specific words.

There are four estimates, E_1, E_2, E_3 and E_4, based respectively on: 1) both words and both tags; 2) $\bar{w}_j$ and the two POS tags; 3) $\bar{w}_{h_j}$ and the two POS tags; 4) the two POS tags alone.

$$E_1 = \tfrac{\eta_1}{\delta_1} \quad E_2 = \tfrac{\eta_2}{\delta_2} \quad E_3 = \tfrac{\eta_3}{\delta_3} \quad E_4 = \tfrac{\eta_4}{\delta_4} \tag{12}$$

where[8]

$$\begin{aligned}
\delta_1 &= C(\langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\delta_2 &= C(\langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\delta_3 &= C(\langle \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\delta_4 &= C(\langle \bar{t}_j \rangle, \langle \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\eta_1 &= C(R_j, \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\eta_2 &= C(R_j, \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\eta_3 &= C(R_j, \langle \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) \\
\eta_4 &= C(R_j, \langle \bar{t}_j \rangle, \langle \bar{t}_{h_j} \rangle, \Delta_{j,h_j})
\end{aligned} \tag{13}$$

[8]

$$C(\langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) = \sum_{x \epsilon \mathcal{V}} C(\langle \bar{w}_j, \bar{t}_j \rangle, \langle x, \bar{t}_{h_j} \rangle, \Delta_{j,h_j})$$

$$C(\langle \bar{t}_j \rangle, \langle \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) = \sum_{x \epsilon \mathcal{V}} \sum_{y \epsilon \mathcal{V}} C(\langle x, \bar{t}_j \rangle, \langle y, \bar{t}_{h_j} \rangle, \Delta_{j,h_j})$$

where $\mathcal{V}$ is the set of all words seen in training data: the other definitions of C follow similarly.

Estimates 2 and 3 compete – for a given pair of words in test data both estimates may exist and they are equally 'specific' to the test case example. (Collins and Brooks 95) suggests the following way of combining them, which favours the estimate appearing more often in training data:

$$E_{23} = \frac{\eta_2 + \eta_3}{\delta_2 + \delta_3} \tag{14}$$

This gives three estimates: E_1, E_{23} and E_4, a similar situation to trigram language modeling for speech recognition (Jelinek 90), where there are trigram, bigram and unigram estimates. (Jelinek 90) describes a deleted interpolation method which combines these estimates to give a 'smooth' estimate, and the model uses a variation of this idea:

If E_1 exists, i.e. $\delta_1 > 0$

$$\hat{F}(R_j \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) = \lambda_1 \times E_1 + (1 - \lambda_1) \times E_{23} \tag{15}$$

Else If E_{23} exists, i.e. $\delta_2 + \delta_3 > 0$

$$\hat{F}(R_j \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) = \lambda_2 \times E_{23} + (1 - \lambda_2) \times E_4 \tag{16}$$

Else

$$\hat{F}(R_j \mid \langle \bar{w}_j, \bar{t}_j \rangle, \langle \bar{w}_{h_j}, \bar{t}_{h_j} \rangle, \Delta_{j,h_j}) = E_4 \tag{17}$$

(Jelinek 90) describes how to find λ values in (15) and (16) which maximise the likelihood of held-out data. We have taken a simpler approach, namely:

$$\begin{aligned}
\lambda_1 &= \frac{\delta_1}{\delta_1 + 1} \\
\lambda_2 &= \frac{\delta_2 + \delta_3}{\delta_2 + \delta_3 + 1}
\end{aligned} \tag{18}$$

These λ values have the desired property of increasing as the denominator of the more 'specific' estimator increases. We think that a proper implementation of deleted interpolation is likely to improve results, although basing estimates on co-occurrence counts alone has the advantage of reduced training times.

2.5 The BaseNP Model

The overall model would be simpler if we could do without the baseNP model and frame everything in terms of dependencies. However the baseNP model is needed for two reasons. First, while adjacency between words is a good indicator of whether there is some relationship between them, this indicator is made substantially stronger if baseNPs are reduced to a single word. Second, it means that words internal to baseNPs are not included in the co-occurrence counts in training data. Otherwise, in a phrase like 'The Securities and Exchange Commission closed yesterday', pre-modifying nouns like 'Securities' and 'Exchange' would be included in co-occurrence counts, when in practice there is no way that they can modify words outside their baseNP.

The baseNP model can be viewed as tagging the gaps between words with $S(tart)$, $C(ontinue)$, $E(nd)$, $B(etween)$ or $N(ull)$ symbols, respectively meaning that the gap is at the start of a $BaseNP$, continues a $BaseNP$, is at the end of a $BaseNP$, is between two adjacent $baseNP$s, or is between two words which are both not in $BaseNPs$. We call the gap before the ith word G_i (a sentence with n words has $n-1$ gaps). For example,

[John Smith] [the president] of [IBM] has announced [his resignation] [yesterday] $\Rightarrow$

John **C** Smith **B** the **C** president **E** of **S** IBM **E** has **N** announced **S** his **C** resignation **B** yesterday

The baseNP model considers the words directly to the left and right of each gap, and whether there is a comma between the two words (we write $c_i = 1$ if there is a comma, $c_i = 0$ otherwise). Probability estimates are based on counts of consecutive pairs of words in **unreduced** training data sentences, where baseNP boundaries define whether gaps fall into the S, C, E, B or N categories. The probability of a baseNP sequence in an unreduced sentence S is then:

$$\prod_{i=2\ldots n} \hat{P}(G_i \mid w_{i-1}, t_{i-1}, w_i, t_i, c_i) \qquad (19)$$

The estimation method is analogous to that described in the sparse data section of this paper. The method is similar to that described in (Ramshaw and Marcus 95; Church 88), where baseNP detection is also framed as a tagging problem.

2.6 Summary of the Model

The probability of a parse tree T, given a sentence S, is:

$$P(T|S) = P(B, D|S) = P(B|S) \times P(D|S, B)$$

The denominator in Equation (9) is not actually constant for different baseNP sequences, but we make this approximation for the sake of efficiency and simplicity. In practice this is a good approximation because most baseNP boundaries are very well defined, so parses which have high enough $P(B|S)$ to be among the highest scoring parses for a sentence tend to have identical or very similar baseNPs. Parses are ranked by the following quantity[9]:

$$\hat{P}(B|S) \times \mathcal{N}(D|S, B) \qquad (20)$$

Equations (19) and (11) define $\hat{P}(B|S)$ and $\mathcal{N}(D|S, B)$. The parser finds the tree which maximises (20) subject to the hard constraint that dependencies cannot cross.

[9] In fact we also model the set of unary productions, U, in the tree, which are of the form $P \rightarrow < C_1 >$. This introduces an additional term, $\hat{P}(U|B, S)$, into (20).

2.7 Some Further Improvements to the Model

This section describes two modifications which improve the model's performance.

- In addition to conditioning on whether dependencies cross commas, a single constraint concerning punctuation is introduced. If for any constituent `Z` in the chart `Z` $\rightarrow$ `<.. X Y ..>` two of its children `X` and `Y` are separated by a comma, then the last word in `Y` must be directly followed by a comma, or must be the last word in the sentence. In training data 96% of commas follow this rule. The rule also has the benefit of improving efficiency by reducing the number of constituents in the chart.
- The model we have described thus far takes the single best sequence of tags from the tagger, and it is clear that there is potential for better integration of the tagger and parser. We have tried two modifications. First, the current estimation methods treat occurrences of the same word with different POS tags as effectively distinct types. Tags can be ignored when lexical information is available by defining

$$C(a, c) = \sum_{b,d \epsilon \mathcal{T}} C(\langle a, b\rangle, \langle c, d\rangle) \qquad (21)$$

where $\mathcal{T}$ is the set of all tags. Hence $C(a, c)$ is the number of times that the words a and c occur in the same sentence, ignoring their tags. The other definitions in (13) are similarly redefined, with POS tags only being used when backing off from lexical information. This makes the parser less sensitive to tagging errors.

Second, for each word w_i the tagger can provide the distribution of tag probabilities $P(t_i|S)$ (given the previous two words are tagged as in the best overall sequence of tags) rather than just the first best tag. The score for a parse in equation (20) then has an additional term, $\prod_{i=1}^{n} P(t_i|S)$, the product of probabilities of the tags which it contains.

Ideally we would like to integrate POS tagging into the parsing model rather than treating it as a separate stage. This is an area for future research.

3 The Parsing Algorithm

The parsing algorithm is a simple bottom-up chart parser. There is no grammar as such, although in practice any dependency with a triple of non-terminals which has not been seen in training data will get zero probability. Thus the parser searches through the space of all trees with non-terminal triples seen in training data. Probabilities of baseNPs in the chart are calculated using (19), while probabilities for other constituents are derived from the dependencies and baseNPs that they contain. A dynamic programming algorithm is used: if two proposed constituents span the same set of words, have the same label, head, and distance from

MODEL	≤ 40 Words (2245 sentences)					≤ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	≤ 2 CBs	LR	LP	CBs	0 CBs	≤ 2 CBs
(1)	84.9%	84.9%	1.32	57.2%	80.8%	84.3%	84.3%	1.53	54.7%	77.8%
(2)	85.4%	85.5%	1.21	58.4%	82.4%	84.8%	84.8%	1.41	55.9%	79.4%
(3)	85.5%	85.7%	1.19	59.5%	82.6%	85.0%	85.1%	1.39	56.8%	79.6%
(4)	85.8%	86.3%	1.14	59.9%	83.6%	85.3%	85.7%	1.32	57.2%	80.8%
SPATTER	84.6%	84.9%	1.26	56.6%	81.4%	84.0%	84.3%	1.46	54.0%	78.8%

Table 3: Results on Section 23 of the WSJ Treebank. **(1)** is the basic model; **(2)** is the basic model with the punctuation rule described in section 2.7; **(3)** is model (2) with POS tags ignored when lexical information is present; **(4)** is model (3) with probability distributions from the POS tagger. **LR/LP** = labeled recall/precision. **CBs** is the average number of crossing brackets per sentence. **0 CBs, ≤ 2 CBs** are the percentage of sentences with 0 or ≤ 2 crossing brackets respectively.

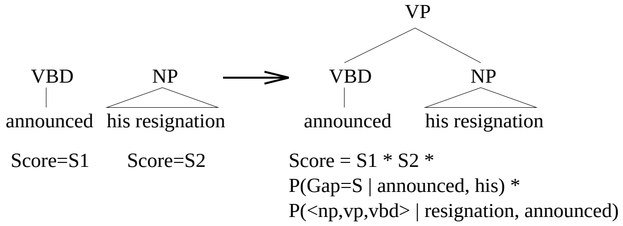

Figure 4: Diagram showing how two constituents join to form a new constituent. Each operation gives two new probability terms: one for the baseNP gap tag between the two constituents, and the other for the dependency between the head words of the two constituents.

Distance Measure	Lexical Information	LR	LP	CBs
Yes	Yes	85.0%	85.1%	1.39
Yes	No	76.1%	76.6%	2.26
No	Yes	80.9%	83.6%	1.51

Table 4: The contribution of various components of the model. The results are for all sentences of ≤ 100 words in section 23 using model (3). For 'no lexical information' all estimates are based on POS tags alone. For 'no distance measure' the distance measure is Question 1 alone (i.e. whether $\bar{w}_j$ precedes or follows $\bar{w}_{h_j}$).

the head to the left and right end of the constituent, then the lower probability constituent can be safely discarded. Figure 4 shows how constituents in the chart combine in a bottom-up manner.

4 Results

The parser was trained on sections 02 - 21 of the Wall Street Journal portion of the Penn Treebank (Marcus et al. 93) (approximately 40,000 sentences), and tested on section 23 (2,416 sentences). For comparison SPATTER (Magerman 95; Jelinek et al. 94) was also tested on section 23. We use the PARSEVAL measures (Black et al. 91) to compare performance:

Labeled Precision $= \frac{number\ of\ correct\ constituents\ in\ proposed\ parse}{number\ of\ constituents\ in\ proposed\ parse}$

Labeled Recall $= \frac{number\ of\ correct\ constituents\ in\ proposed\ parse}{number\ of\ constituents\ in\ treebank\ parse}$

Crossing Brackets = number of constituents which violate constituent boundaries with a constituent in the treebank parse.

For a constituent to be 'correct' it must span the same set of words (ignoring punctuation, i.e. all tokens tagged as commas, colons or quotes) and have the same label[10] as a constituent in the treebank parse. Four configurations of the parser were tested: **(1)** The basic model; **(2)** The basic model with the punctuation rule described in section 2.7; **(3)** Model (2) with tags ignored when lexical information is present, as described in 2.7; and **(4)** Model (3) also using the full probability distributions for POS tags. We should emphasise that test data outside of section 23 was used for all development of the model, avoiding the danger of implicit training on section 23. Table 3 shows the results of the tests. Table 4 shows results which indicate how different parts of the system contribute to performance.

4.1 Performance Issues

All tests were made on a Sun SPARCServer 1000E, using 100% of a 60Mhz SuperSPARC processor. The parser uses around 180 megabytes of memory, and training on 40,000 sentences (essentially extracting the co-occurrence counts from the corpus) takes under 15 minutes. Loading the hash table of bigram counts into memory takes approximately 8 minutes.

Two strategies are employed to improve parsing efficiency. First, a constant probability threshold is used while building the chart – any constituents with lower probability than this threshold are discarded. If a parse is found, it must be the highest ranked parse by the model (as all constituents discarded have lower probabilities than this parse and could

[10] SPATTER collapses `ADVP` and `PRT` to the same label, for comparison we also removed this distinction when calculating scores.

not, therefore, be part of a higher probability parse). If no parse is found, the threshold is lowered and parsing is attempted again. The process continues until a parse is found.

Second, a beam search strategy is used. For each span of words in the sentence the probability, P_h, of the highest probability constituent is recorded. All other constituents spanning the same words must have probability greater than $\frac{P_h}{\beta}$ for some constant beam size β – constituents which fall out of this beam are discarded. The method risks introducing search-errors, but in practice efficiency can be greatly improved with virtually no loss of accuracy. Table 5 shows the trade-off between speed and accuracy as the beam is narrowed.

Beam Size β	Speed Sentences/minute	LR	LP	CBs
1000	118	84.9%	85.1%	1.39
150	166	84.8%	85.1%	1.38
20	217	84.7%	85.0%	1.40
3	261	84.1%	84.5%	1.44
1.5	283	83.7%	84.1%	1.48
1.2	289	83.5%	83.9%	1.50

Table 5: The trade-off between speed and accuracy as the beam-size is varied. Model (3) was used for this test on all sentences ≤ 100 words in section 23.

5 Conclusions and Future Work

We have shown that a simple statistical model based on dependencies between words can parse Wall Street Journal news text with high accuracy. The method is equally applicable to tree or dependency representations of syntactic structures.

There are many possibilities for improvement, which is encouraging. More sophisticated estimation techniques such as deleted interpolation should be tried. Estimates based on relaxing the distance measure could also be used for smoothing – at present we only back-off on words. The distance measure could be extended to capture more context, such as other words or tags in the sentence. Finally, the model makes no account of valency.

Acknowledgements

I would like to thank Mitch Marcus, Jason Eisner, Dan Melamed and Adwait Ratnaparkhi for many useful discussions, and for comments on earlier versions of this paper. I would also like to thank David Magerman for his help with testing SPATTER.

References

E. Black et al. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop.*

T. Briscoe and J. Carroll. 1993. Generalized LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. *Computational Linguistics*, 19(1):25-60.

K. Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Second Conference on Applied Natural Language Processing, ACL.*

M. Collins and J. Brooks. 1995. Prepositional Phrase Attachment through a Backed-off Model. *Proceedings of the Third Workshop on Very Large Corpora*, pages 27-38.

D. Hindle and M. Rooth. 1993. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19(1):103-120.

F. Jelinek. 1990. Self-organized Language Modeling for Speech Recognition. In *Readings in Speech Recognition.* Edited by Waibel and Lee. Morgan Kaufmann Publishers.

F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, S. Roukos. 1994. Decision Tree Parsing using a Hidden Derivation Model. *Proceedings of the 1994 Human Language Technology Workshop*, pages 272-277.

J. Lafferty, D. Sleator and, D. Temperley. 1992. Grammatical Trigrams: A Probabilistic Model of Link Grammar. *Proceedings of the 1992 AAAI Fall Symposium on Probabilistic Approaches to Natural Language.*

D. Magerman. 1995. Statistical Decision-Tree Models for Parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276-283.

D. Magerman and M. Marcus. 1991. Pearl: A Probabilistic Chart Parser. *Proceedings of the 1991 European ACL Conference*, Berlin, Germany.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.

F. Pereira and Y. Schabes. 1992. Inside-Outside Reestimation from Partially Bracketed Corpora. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128-135.

L. Ramshaw and M. Marcus. 1995. Text Chunking using Transformation-Based Learning. *Proceedings of the Third Workshop on Very Large Corpora*, pages 82-94.

A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. *Conference on Empirical Methods in Natural Language Processing*, May 1996.

M. M. Wood. 1993. *Categorial Grammars*, Routledge.